GStreamer in the Medical Simulation Environment

A project's journey from Flash to WebRTC, from a monolithic pipeline to multiple pipelines.

Jeff Wilson

Thomas Goodwin

Laerdal

One goal. One mission.




Laerdal
helping save lives

Laerdal
helping save lives

Laerdal
Foundation

Laerdal
Million Lives Fund

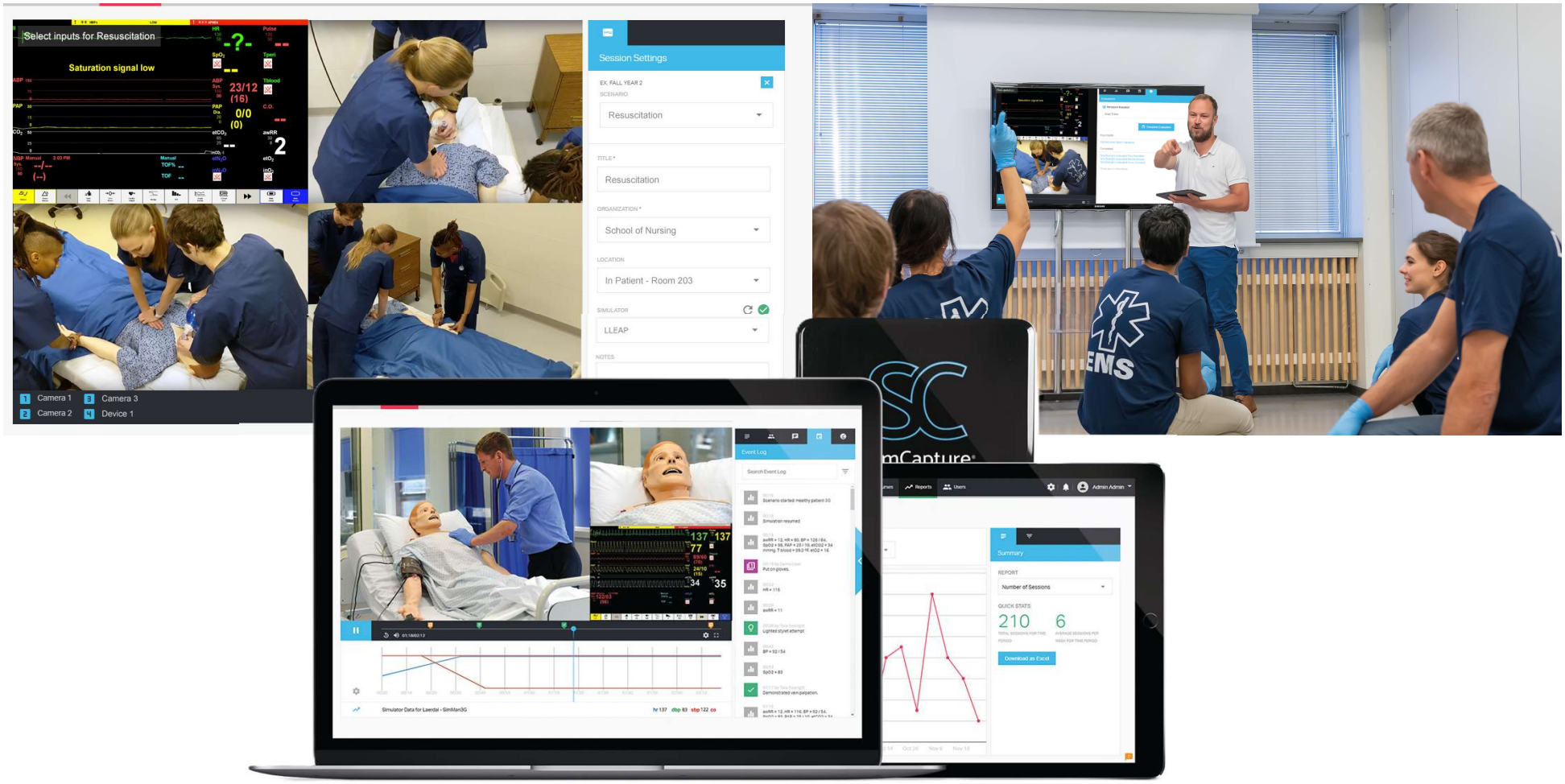
Laerdal Medical is a world leader in healthcare education and resuscitation training. By using immersive technologies and data-centric insights we can help to increase survival and improve healthcare quality. We empower lifesavers, and healthcare workers, to help them save more lives.

SimCapture: CaptureNode

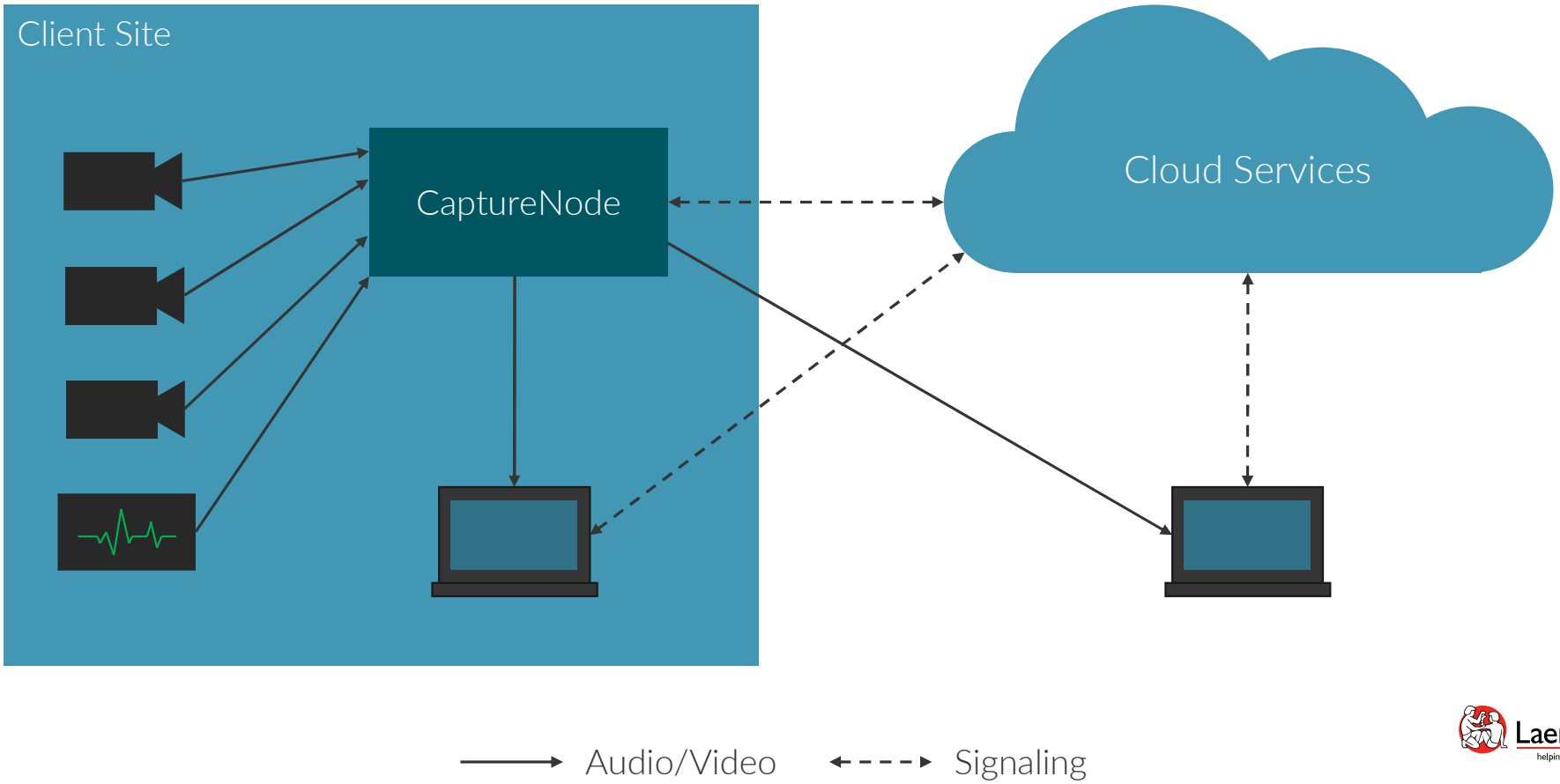
1. SimCapture Introduction
2. CaptureNode's progression from Flash to WebRTC
3. Incorporating GStreamer for various sources
4. The progression from a monolithic pipeline to multiple pipelines via inters
5. Our First Plugin and Elements
6. Writing MP4s to S3 During Recordings
7. Handling Transient Sources
8. ONVIF Device Manager
9. Pain Point: C++ and GObject

Laerdal
helping save lives

SimCapture


Select inputs for Resuscitation
Saturation signal low
Session Settings
Resuscitation
School of Nursing
In Patient - Room 203
LLEAP
Camera 1
Camera 2
Camera 3
Device 1
Event Log
Summary
210
6
Download as Excel

SimCapture

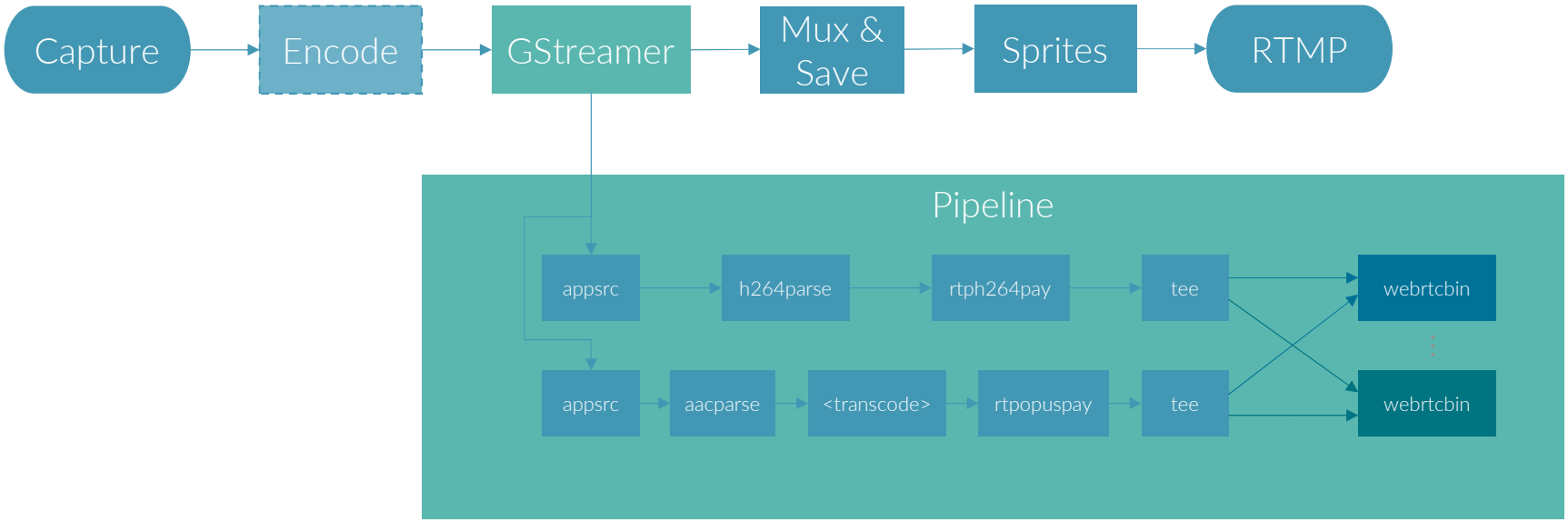


Laerdal
helping save lives

CaptureNode: Flash Streaming

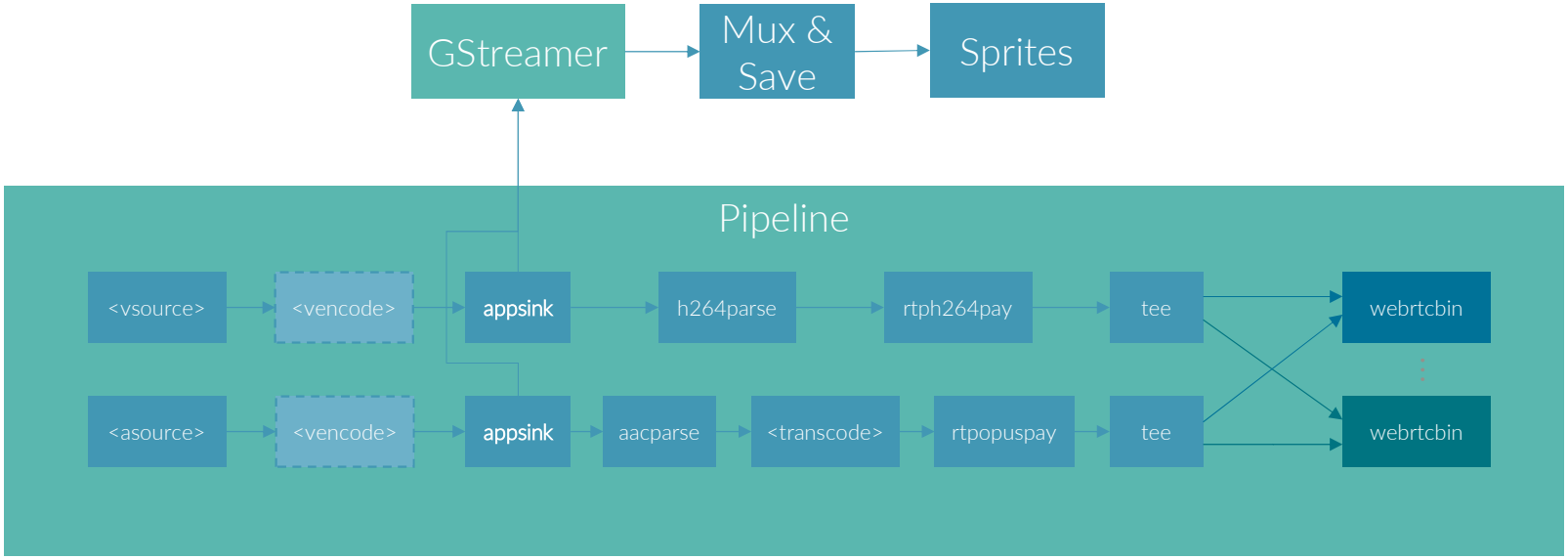
Laerdal
helping save lives

CaptureNode: Flash to WebRTC



Laerdal
helping save lives

CaptureNode: GStreamer Sources



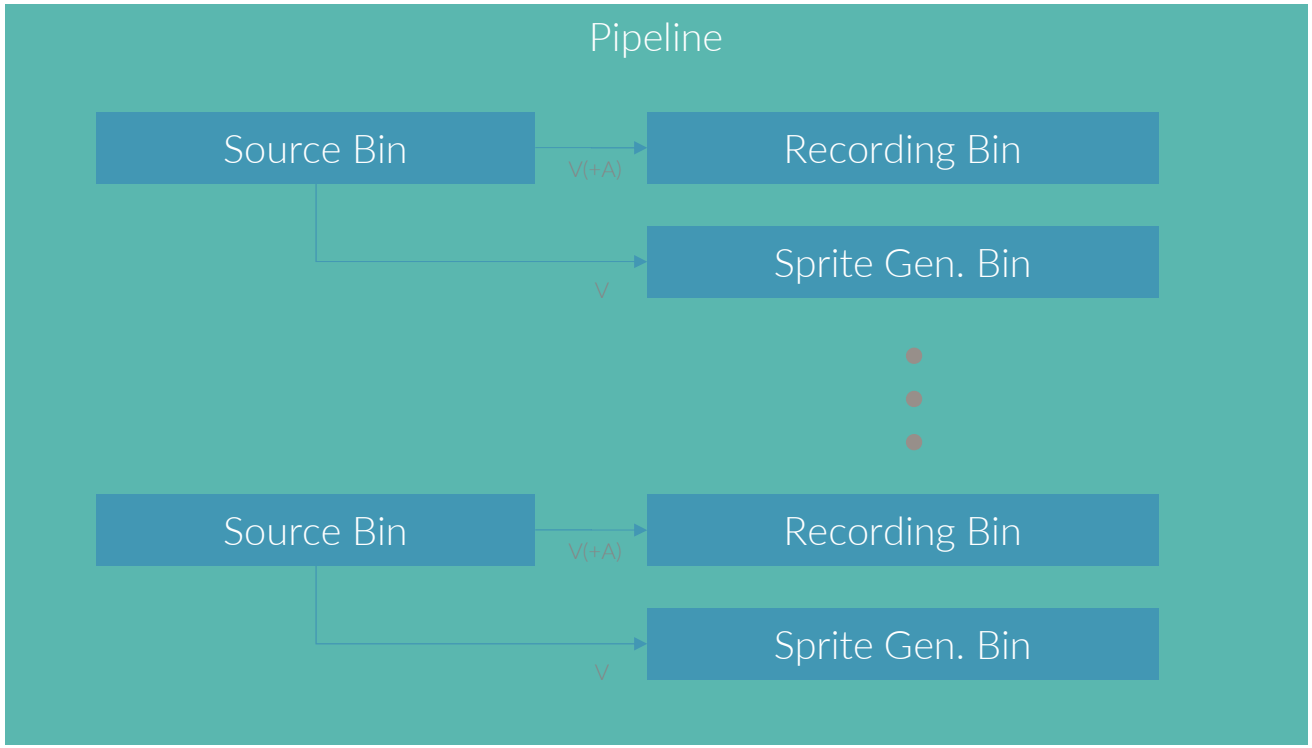
Laerdal
helping save lives

CaptureNode: GStreamer Sources

- Most common input sources:
  - Network security cameras & frame grabbers
  - USB cameras & frame grabbers
  - Custom screen recorder application
- Other sources exist such as NDI
- Device configuration & discovery written in C# while capture written in C/C++
- Mixing between GStreamer sources and legacy sources caused some issues
- Re-acquiring inputs that were lost is complicated

Laerdal
helping save lives

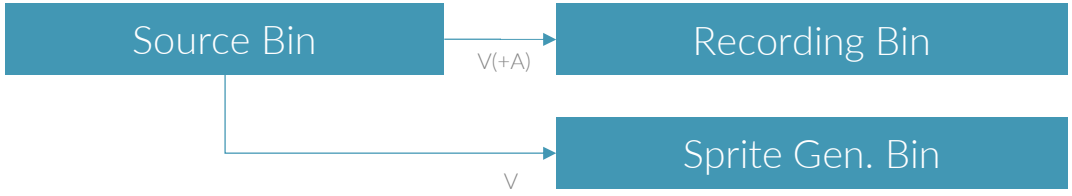
From Monolithic to Multiple Pipelines

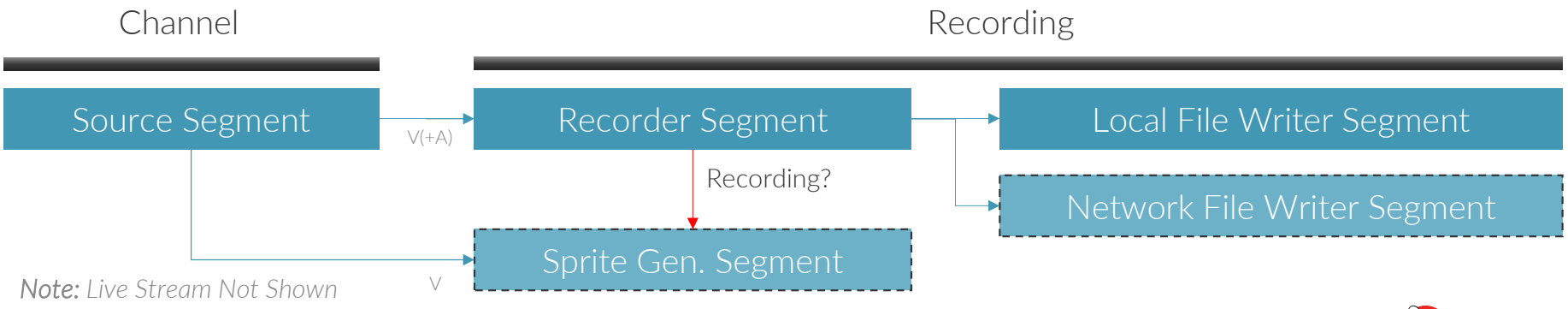


Note: Live Stream Not Shown

Laerdal
helping save lives

From Monolithic to Multiple Pipelines

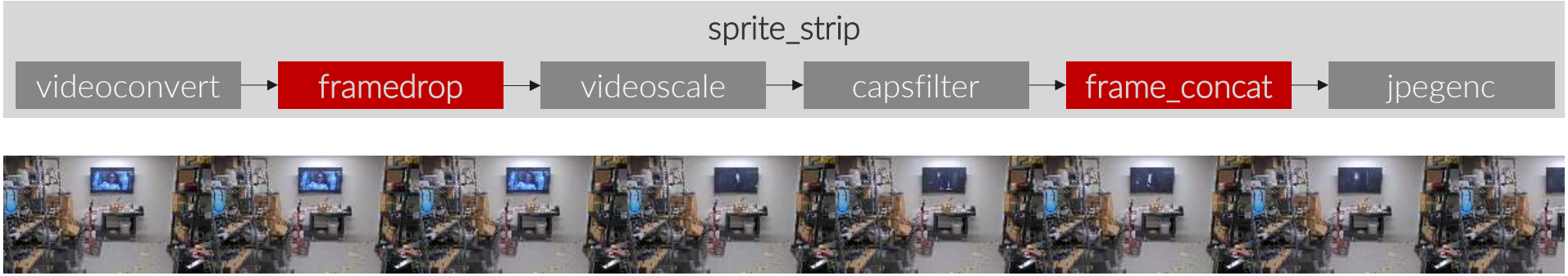




Laerdal
helping save lives

Our First GStreamer Plugin

- Simple concept:
  - Sample a raw video feed down to an arbitrarily low frame rate.
  - Concatenate the frames according to scale requirements
  - Output the resulting JPEG
- As first experiences go, this went well.
  - Tried *videorate* element but needed something simpler: *framedrop*.
  - Gained experience with GstCheck/GstHarness

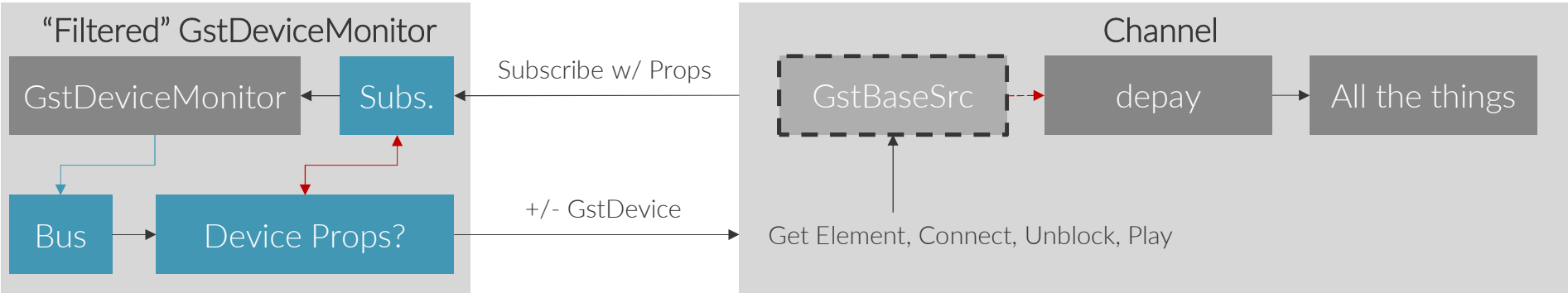


Laerdal
helping save lives

Handling Transient Sources

Situation:

A source device element streams into the channel processing pipeline, eventually encountering a connectivity problem. GstBaseSrc emits EOS, pad task stops. You handled the EOS, saving the rest of the pipeline, but toggling GstState of the source element does not resume the feed later. Now what?



Solution:

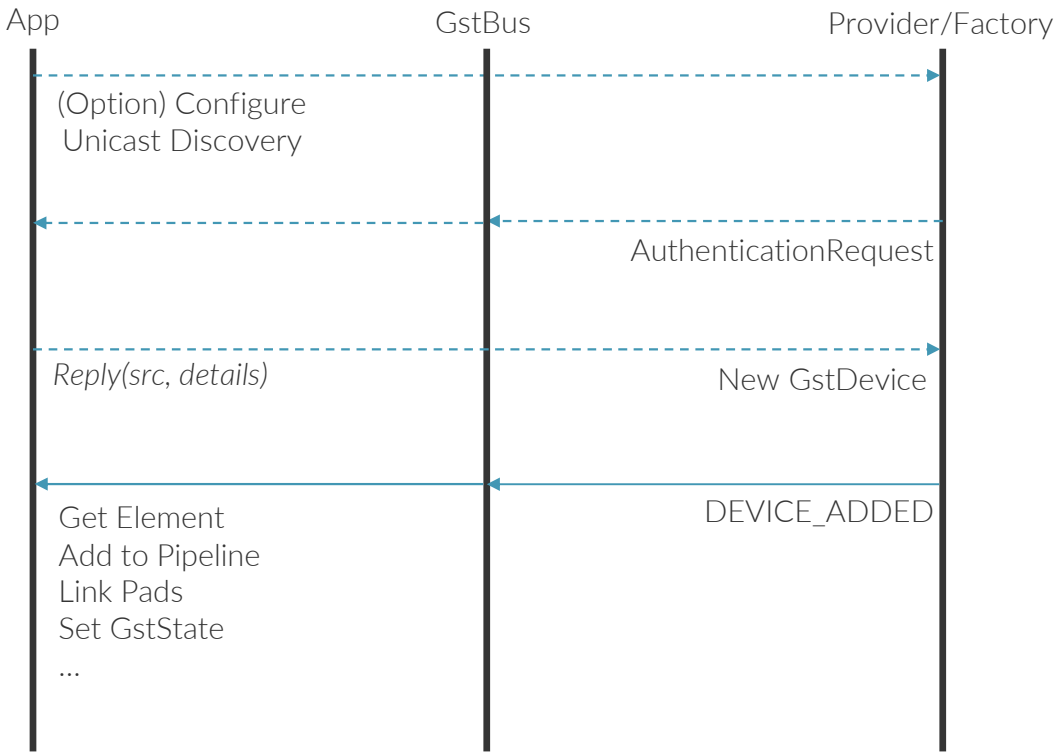
A Channel is assigned some device-specific identifiers. It subscribes to the monitor wrapper with both the usual filter API and those properties. When the GstDeviceMonitor emits Device Added on the bus, the wrapper inspects the device properties, compares to the subscriber list. If there is a match, the Channel is notified. The channel inserts the Device's element, unblocks, and resumes the stream.

Laerdal
helping save lives

ONVIF Device Monitor

Hey, let's replace capture inputs with FilteredGstDeviceMonitor!

- Began as a hackathon project
- Leverage existing ONVIF-RS Project, extend our knowledge of GStreamer+Rust
- Use WS-Discovery to detect devices on the network
- Define a new authentication API for DeviceMonitor's GstBus



Laerdal
helping save lives

Writing MP4s to AWS S3

1. Trimmed-down from our fork of the AWS S3 Sink C++ Element, implemented seek operations
    1. Cache hit is a direct seek on the memory heap
    2. Cache miss is an S3 download, seek, upload
2. Ported to Rust with the (1.1) cache hit logic (gst-plugins-rs !1641)
    1. N=0, default behavior
    2. N>=1, cache the first N parts
    3. N<=-1, cache the tail N parts

Thank you: Arun Raghavan!

Laerdal
helping save lives

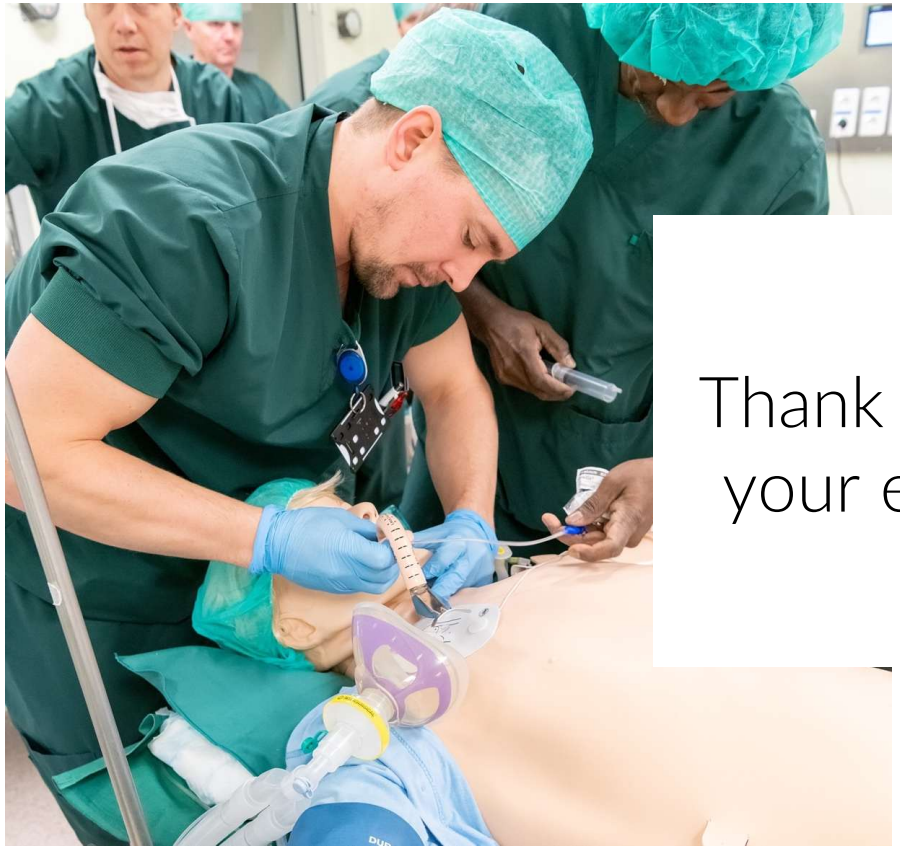
*C++ Variant: https://github.com/blinemedical/amazon-s3-gst-plugin

Pain Point: C++ and GObject

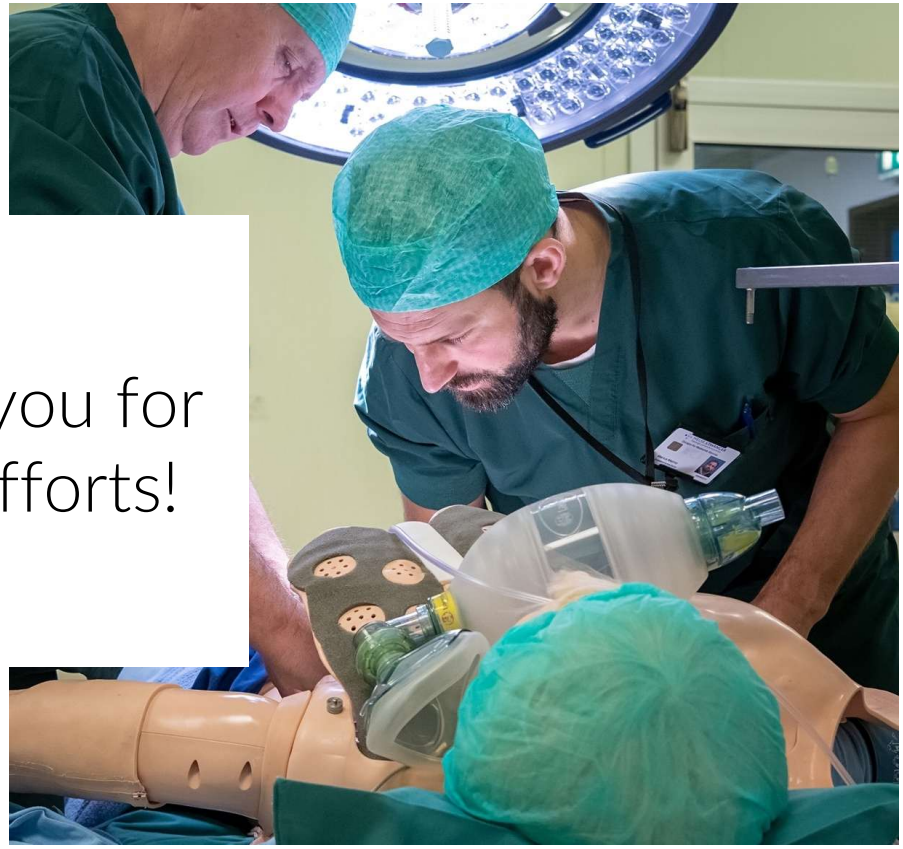
Lack of direct integration paths to synchronize memory management models can result in access.

- The obvious fix is addressing the reference counting issue but tracing the root cause can be daunting.
- Usual indication of a problem: callbacks and signals needing static methods or other mechanisms like std::function pointers on the heap. When synchronization breaks, you get AVs here.
- One work-around we use is `WeakReferenceWrapper<T, P>`:
  - `T` -> `GstElement`, `GstBus`, etc.
  - `P` -> `std::unique_ptr<T, GObjectFunctor<T>>`
    where `GObjectFunctor` is a `struct operator()` for unreffing the internal ref-counted pointer
  - Mutex-guarded raw pointer maintained by `g_object_weak_[ref/unref]`
  - Public `Set()/Clear()` for modifying the raw pointer
  - Public `AsRef()` to get a reference-counted unique pointer scoped to the caller
  - Protected virtual method for *"I've been destroyed."*

Looking forward to a talk about a third-party solution later which shows some promise!



Thank you for your efforts!



Links

- SimCapture
- `awss3sink` seeking MR
- `gst-laerdallabs-rs` – ONVIF discovery


Laerdal
helping save lives